**Code:**

#include <iostream>

using namespace std;

int main()

{

cout << "Enter Size of Stack: ";

int n;

cin >> n;

int arr[n];

int top = -1;

int op;

cout << "1. Push\n2. Pop\n3. Top\n0. Exit\n";

while (1)

{

cout << "Enter Option: ";

cin >> op;

switch (op)

{

case 0:

return 0;

case 1:

{

if (top == n - 1)

{

cout << "Stack is Full." << endl;

break;

}

int val;

cout << "Enter Value: ";

cin >> val;

arr[++top] = val;

}

break;

case 2:

{

if (top == -1)

{

cout << "Stack is Empty." << endl;

break;

}

--top;

}

break;

case 3:

{

if (top == -1)

{

cout << "Stack is Empty." << endl;

break;

}

cout << "Top: " << arr[top] << endl;

}

break;

break;

default:

cout << "You Entered Wrong Option! Try again." << endl;

break;

}

}

}

**Stack Output:**

Enter Size of Stack: 2

1. Push

2. Pop

3. Top

0. Exit

Enter Option: 2

Stack is Empty.

Enter Option: 1

Enter Value: 10

Enter Option: 1

Enter Value: 11

Enter Option: 3

Top: 11

Enter Option: 1

Stack is Full.

Enter Option: 2

Enter Option: 3

Top: 10

Enter Option: 2

Enter Option: 2

Stack is Empty.

Enter Option: 0

**Code:**

**#include <iostream>**

using namespace std;

void bubbleSort(*string* &*arr*, int *n*)

{

char t;

for (int i = 0; i < *n* - 1; i++)

{

for (int j = 0; j < *n* - i - 1; j++)

{

if (*arr*[j] > *arr*[j + 1])

{

t = *arr*[j + 1];

*arr*[j + 1] = *arr*[j];

*arr*[j] = t;

}

}

}

}

int main()

{

cout << "Enter Size of Array: ";

int n;

cin >> n;

*string* arr;

cout << "Enter String: " << endl;

cin >> arr;

bubbleSort(arr, n);

cout << "Sorted Array" << endl;

for (int i = 0; i < n; i++)

{

cout << arr[i] << " ";

}

cout << endl;

return 0;

}

**Bubble Sort Output:**

Enter Size of Array: 5

Enter Elements of the array:

3 2 5 1 2

Sorted Array

1 2 2 3 5

**#include <bits/stdc++.h>**

using namespace std;

void arrayInput(int *arr*[], int *n*)

{

for (int i = 0; i < *n*; i++)

cin >> *arr*[i];

}

void print(int *arr*[], int *n*)

{

for (int i = 0; i < *n*; i++)

cout << *arr*[i] << " ";

cout << endl;

}

int binarySearch(int *arr*[], int *l*, int *h*, int *key*)

{

if (*l* > *h*)

return 0;

int mid = (*l* + *h*) / 2;

if (*arr*[mid] == *key*)

return mid + 1;

if (*arr*[mid] < *key*)

return binarySearch(*arr*, mid + 1, *h*, *key*);

else

return binarySearch(*arr*, *l*, mid - 1, *key*);

}

int main()

{

cout << "Enter Size of Array: ";

int n;

cin >> n;

int arr[n];

cout << "Enter Elements of the array: " << endl;

arrayInput(arr, n);

sort(arr, arr+n);

int key;

cout << "Enter Key: ";

cin >> key;

int pos = binarySearch(arr, 0, n - 1, key);

if (pos)

cout << "Your key found." << endl;

else

cout << "Your key is not found." << endl;

return 0;

}

**Binary Search:**

Enter Size of Array: 5

Enter Elements of the array:

3 2 4 1 5

Enter Key: 5

Your key found.

**#include<iostream>**

using namespace std;

void arrayInput(int *arr*[], int *n*){

for(int i=0;i<n;i++)cin>>arr[i];

}

void print(int *arr*[],int *n*){

for(int i=0;i<n;i++)cout<<arr[i]<<" ";

cout<<endl;

}

void marge(int *arr*[], int *l*, int *mid*,int *r*){

int te[r-l+1];

int i=l;

int k=0;

int j=mid+1;

while(i<=mid and j<=r){

if(arr[i]<arr[j]){

te[k++]=arr[i++];

continue;

}else te[k++]=arr[j++];

}

while(i<=mid)te[k++]=arr[i++];

while(j<=r)te[k++]=arr[j++];

for(int i=0;i<k;i++){

arr[l+i]=te[i];

}

}

void margeSort(int *arr*[],int *l*, int *r*){

if(l>=r)return;

int mid =(l+r)/2;

margeSort(arr,l,mid);

margeSort(arr,mid+1,r);

marge(arr,l,mid,r);

}

int main(){

cout<<"Enter Size of Array: ";

int n;

cin>>n;

int arr[n];

cout<<"Enter Elements of the array: "<<endl;

arrayInput(arr,n);

margeSort(arr,0,n-1);

cout<<"Sorted Array"<<endl;

print(arr,n);

return 0;

}

**Merge Sort:**

Enter Size of Array: 5

Enter Elements of the array:

3 2 5 1 4

Sorted Array

1 2 3 4 5

**#include<iostream>**

using namespace std;

void arrayInput(int *arr*[], int *n*){

for(int i=0;i<*n*;i++)cin>>*arr*[i];

}

void print(int *arr*[],int *n*){

for(int i=0;i<*n*;i++)cout<<*arr*[i]<<" ";

cout<<endl;

}

void selectionSort(int *arr*[],int *size*){

int t;

for(int i=0;i<*size*-1;i++){

int j =i+1;

int k=i;

while(j<*size*){

if(*arr*[k]>*arr*[j]){

k = j;

}

j++;

}

t = *arr*[k];

*arr*[k] = *arr*[i];

*arr*[i] = t;

}

}

int main(){

cout<<"Enter Size of Array: ";

int n;

cin>>n;

int arr[n];

cout<<"Enter Elements of the array: "<<endl;

arrayInput(arr,n);

selectionSort(arr,n);

cout<<"Sorted Array"<<endl;

print(arr,n);

return 0;

}

**Selection Sort:**

Enter Size of Array: 5

Enter Elements of the array:

2 4 3 1 5

Sorted Array

1 2 3 4 5

**#include<iostream>**

using namespace std;

void arrayInput(int *arr*[], int *n*){

for(int i=0;i<*n*;i++)cin>>*arr*[i];

}

void print(int *arr*[],int *n*){

for(int i=0;i<*n*;i++)cout<<*arr*[i]<<" ";

cout<<endl;

}

void insertionSort(int *arr*[],int *n*){

for(int i=1;i<*n*;i++){

int k = *arr*[i];

int j = i-1;

while(j>=0 and *arr*[j]>k){

*arr*[j+1]=*arr*[j];

j--;

}

*arr*[j+1]= k;

}

}

int main(){

cout<<"Enter Size of Array: ";

int n;

cin>>n;

int arr[n];

cout<<"Enter Elements of the array: "<<endl;

arrayInput(arr,n);

insertionSort(arr,n);

cout<<"Sorted Array"<<endl;

print(arr,n);

return 0;

}

**Insertion Sort:**

Enter Size of Array: 5

Enter Elements of the array:

4 3 5 1 2

Sorted Array

1 2 3 4 5

**#include <iostream>**

using namespace std;

typedef struct *Node*

{

int data;

*Node* \*l;

*Node* \*r;

Node(int *d*)

{

data = *d*;

l = r = NULL;

}

} *Node*;

*Node* \*insertBST(*Node* \**root*, int *data*)

{

if (*root* == NULL)

return **new** *Node*(*data*);

if (*root*->data > *data*)

*root*->l = insertBST(*root*->l, *data*);

else

*root*->r = insertBST(*root*->r, *data*);

return *root*;

}

int search(*Node* \**root*, int *key*)

{

if (*root* == NULL)

return -1;

if (*root*->data == *key*)

return 1;

if (*root*->data > *key*)

return search(*root*->l, *key*);

else

return search(*root*->r, *key*);

}

void inorder(*Node* \**root*)

{

if (*root* == NULL)

return;

inorder(*root*->l);

cout << *root*->data << " ";

inorder(*root*->r);

}

int main()

{

*Node* \*root = NULL;

cout << "Enter How many element do you want: ";

int a;

cin >> a;

int v;

cout << "Enter Elements: ";

cin >> v;

root = insertBST(root, v);

for (int i = 0; i < a - 1; i++)

{

cin >> v;

insertBST(root, v);

}

int key;

cout << "Enter Key: ";

cin >> key;

if (search(root, key) != -1)

cout << "Found" << endl;

else

cout << "Not Found" << endl;

return 0;

}

**Binary Search Tree:**

Enter How many element do you want: 5

Enter Elements: 4 35 3 6 3

Enter Key: 7

Not Found

**#include <iostream>**

using namespace std;

int main()

{

cout << "Enter Size of Queue: ";

int n;

cin >> n;

int arr[n];

int front = -1, rare = -1;

int op;

cout << "1. Enqueue\n2. Dequeuq\n3. Front\n0. Exit\n";

while (1)

{

cout << "Enter Option: ";

cin >> op;

switch (op)

{

case 0:

return 0;

case 1:

{

if (rare == n - 1)

{

cout << "Queue is Full." << endl;

break;

}

int val;

cout << "Enter Value: ";

cin >> val;

arr[++rare] = val;

if (front == -1)

front++;

}

break;

case 2:

{

if (front == -1 or front > rare)

{

cout << "Queue is Empty." << endl;

break;

}

front++;

}

break;

case 3:

{

if (front == -1 or front > rare)

{

cout << "Queue is Empty." << endl;

break;

}

cout << "Front: " << arr[front] << endl;

}

break;

break;

default:

cout << "You Entered Wrong Option! Try again." << endl;

break;

}

}

}

**Queue:**

Enter Size of Queue: 3

1. Enqueue

2. Dequeuq

3. Front

0. Exit

Enter Option: 1

Enter Value: 2

Enter Option: 1

Enter Value: 3

Enter Option: 3

Front: 2

Enter Option: 2

Enter Option: 3

Front: 3

Enter Option: 0

#include <stdio.h>

#include <stdlib.h>

struct *node*

{

int data;

struct *node* \*next;

};

void insert(struct *node* \*\**pointer*, int *n*)

{

struct *node* \*newN;

newN = malloc(sizeof(struct *node*));

struct *node* \*temp = \**pointer*;

newN->data = *n*;

newN->next = NULL;

if (\**pointer* == NULL)

{

\**pointer* = newN;

return;

}

while (temp->next != NULL)

temp = temp->next;

temp->next = newN;

return;

}

void delete (struct *node* \**del*, int *p*)

{

int cout = 0;

struct *node* \*temp;

while (*del* != NULL)

{

cout++;

if (cout == *p*)

break;

temp = *del*;

*del* = *del*->next;

}

struct *node* \*temp2 = *del*;

*del* = *del*->next;

temp->next = *del*;

free(temp2);

}

void insertpos(struct *node* \*\**ins*, int *p*, int *n*)

{

if (*p* < 1)

printf("Position invalid ");

else

{

while (*p*--)

{

if (*p* == 0)

{

struct *node* \*newN;

newN = malloc(sizeof(struct *node*));

newN->data = *n*;

newN->next = NULL;

struct *node* \*temp = newN;

temp->next = \**ins*;

\**ins* = temp;

}

//\*ins=\*ins->next

*ins* = &(\**ins*)->next;

}

}

}

void printList(struct *node* \**p*)

{

printf("Current list is : ");

while (*p* != NULL)

{

printf("%d ", *p*->data);

*p* = *p*->next;

}

}

int main()

{

struct *node* \*head = NULL;

int option;

Start:

printf("\nSelect option : 1.Insert in the end, 2.Insert in position, 3.Delete position ,4.Print, Press 0 to Exit\n");

scanf("%d", &option);

if (option == 1)

{

int value;

printf("Enter the insert value : ");

scanf("%d", &value);

insert(&head, value);

goto Start;

}

else if (option == 3)

{

int pos;

printf("Enter delete position : ");

scanf("%d", &pos);

delete (head, pos);

goto Start;

}

else if (option == 2)

{

int po, value;

printf("ENter the position and value : ");

scanf("%d%d", &po, &value);

insertpos(&head, po, value);

goto Start;

}

else if (option == 4)

{

printList(head);

goto Start;

}

return 0;

}

**Linked List:**

Select option : 1.Insert in the end, 2.Insert in position, 3.Delete position ,4.Print, Press 0 to Exit

1

Enter the insert value : 5

Select option : 1.Insert in the end, 2.Insert in position, 3.Delete position ,4.Print, Press 0 to Exit

1

Enter the insert value : 35

Select option : 1.Insert in the end, 2.Insert in position, 3.Delete position ,4.Print, Press 0 to Exit

4

Current list is : 5 35

Select option : 1.Insert in the end, 2.Insert in position, 3.Delete position ,4.Print, Press 0 to Exit

3

Enter delete position : 2

Select option : 1.Insert in the end, 2.Insert in position, 3.Delete position ,4.Print, Press 0 to Exit

4

Current list is : 5

Select option : 1.Insert in the end, 2.Insert in position, 3.Delete position ,4.Print, Press 0 to Exit

0

#include <iostream>

using namespace std;

void arrayInput(int *arr*[], int *n*)

{

for (int i = 0; i < *n*; i++)

cin >> *arr*[i];

}

int search(int *arr*[], int *n*, int *key*)

{

for (int i = 0; i < *n*; i++)

{

if (*key* == *arr*[i])

return i + 1;

}

return 0;

}

void insert(int *arr*[], int *n*, int *pos*, int *val*)

{

int i = *n* - 1;

while (*pos* <= i)

{

*arr*[i + 1] = *arr*[i];

i--;

}

*arr*[*pos*] = *val*;

}

void del(int *arr*[], int *n*, int *pos*)

{

int i = *pos*;

while (i < *n* - 1)

{

*arr*[i] = *arr*[i + 1];

i++;

}

}

void print(int *arr*[], int *n*)

{

for (int i = 0; i < *n*; i++)

cout << *arr*[i] << " ";

cout << endl;

}

int main()

{

int arr[1000];

cout << "Enter Size of Array: ";

int n;

cin >> n;

cout << "Enter Elements of the array: " << endl;

arrayInput(arr, n);

int op;

cout << "1. Insert\n2. Delete\n3. Search\n4. Travers\n0. Exit\n";

while (1)

{

cout << "Enter Option: ";

cin >> op;

switch (op)

{

case 0:

return 0;

case 1:

{

cout << "Enter position: ";

int pos, val;

cin >> pos;

cout << "Enter Value: ";

cin >> val;

pos--;

insert(arr, n, pos, val);

n++;

}

break;

case 2:

{

cout << "Enter position: ";

int pos;

cin >> pos;

pos--;

del(arr, n, pos);

n--;

}

break;

case 3:

{

cout << "Enter Key: ";

int key;

cin >> key;

int pos = search(arr, n, key);

if (pos)

cout << "Your key found at position " << pos << endl;

else

cout << "Your key is not found." << endl;

}

break;

case 4:

print(arr, n);

break;

default:

cout << "You Entered Wrong Option! Try again." << endl;

break;

}

}

}

**ARRAY:**

Enter Size of Array: 5

Enter Elements of the array:

4 5 2 4 5

1. Insert

2. Delete

3. Search

4. Travers

0. Exit

Enter Option: 1

Enter position: 4

Enter Value: 5

Enter Option: 4

4 5 2 5 4 5

Enter Option: 2

Enter position: 1

Enter Option: 4

5 2 5 4 5

Enter Option: 3

Enter Key: 2

Your key found at position 2

Enter Option: 0

**Code :**

#include <iostream>

using namespace std;

class *Matrix*

{

public:

int arr[100][100];

int r, c;

Matrix(int *row*, int *col*)

{

r = *row*;

c = *col*;

}

void input()

{

for (int i = 0; i < r; i++)

for (int j = 0; j < c; j++)

cin >> arr[i][j];

}

void print()

{

for (int i = 0; i < r; i++)

{

for (int j = 0; j < c; j++)

{

cout << arr[i][j] << " ";

}

cout << endl;

}

}

*Matrix* operator\*(*Matrix* &*b*)

{

*Matrix* w(r, *b*.c);

for (int i = 0; i < r; i++)

{

for (int j = 0; j < *b*.c; j++)

{

w.arr[i][j] = 0;

}

}

for (int i = 0; i < r; i++)

{

for (int j = 0; j < *b*.c; j++)

{

for (int k = 0; k < c; k++)

{

w.arr[i][j] += arr[i][k] \* *b*.arr[k][j];

}

}

}

return w;

}

*Matrix* operator+(*Matrix* &*b*)

{

*Matrix* w(r, c);

for (int i = 0; i < r; i++)

{

for (int j = 0; j < c; j++)

{

w.arr[i][j] = arr[i][j] + *b*.arr[i][j];

}

}

return w;

}

*Matrix* t()

{

*Matrix* w(c, r);

for (int i = 0; i < c; i++)

{

for (int j = 0; j < r; j++)

{

w.arr[i][j] = arr[j][i];

}

}

return w;

}

};

int main()

{

cout << "1. Sum\n2. Multiply\n 3.Transpose\n0. Exit";

int op;

while (1)

{

cout << "Enter option: ";

cin >> op;

switch (op)

{

case 1:

{

cout << "Enter Size of Matrix A:";

int x, y;

cin >> x >> y;

*Matrix* A(x, y);

cout << "Enter Size of Matrix B:";

cin >> x >> y;

*Matrix* B(x, y);

if (A.r == B.r && A.c == B.c)

{

cout << "Enter Matrix A:" << endl;

A.input();

cout << "Enter Matrix B:" << endl;

B.input();

*Matrix* C = A + B;

C.print();

}

else

{

cout << "Summation Not posible." << endl;

}

}

break;

case 2:

{

cout << "Enter Size of Matrix A:";

int x, y;

cin >> x >> y;

*Matrix* A(x, y);

cout << "Enter Size of Matrix B:";

cin >> x >> y;

*Matrix* B(x, y);

if (A.c == B.r)

{

cout << "Enter Matrix A:" << endl;

A.input();

cout << "Enter Matrix B:" << endl;

B.input();

*Matrix* C = A \* B;

C.print();

}

else

{

cout << "Multiplication Not posible." << endl;

}

}

break;

case 3:

{

cout << "Enter Size of Matrix A:";

int x, y;

cin >> x >> y;

*Matrix* A(x, y);

cout << "Enter Matrix A:" << endl;

A.input();

*Matrix* C = A.t();

C.print();

}

break;

default:

return 0;

}

}

}

**Matrix Output:**

1. Sum

2. Multiply

3.Transpose

0. Exit

Enter option: 1

Enter Size of Matrix A:2

2

Enter Size of Matrix B:2 2

Enter Matrix A:

1 2 3 4

Enter Matrix B:

4 3 2 1

5 5

5 5

Enter option: 2

Enter Size of Matrix A:2 2

Enter Size of Matrix B:2 2

Enter Matrix A:

1 2 3 4

Enter Matrix B:

4 3 2 1

8 5

20 13

Enter option: 0